**Notes**

**Room Db Creation**

1. Create entity class that represents a table, Then create entity columns with it’s getters and setters
2. Create Dao Class ”Data access objects ” with crud operations, any custom query annotate it as query and write the query
3. Create RoomDatabase
   * It must be abstract class
   * It must extend RoomDatabase
   * It must be a singletone
   * Annotate it with database that takes
     + array of entities
     + Version number that must be increased on any update to database structure
4. Synchronized means one thread at a time can access this singleton object in case of multithreading
5. Repository class provide an abstraction layer between data layer and rest of application it provides data from multiple sources
   * We create the database in it’s constrctor and init dao object and get all notes liveData List
   * Then we provide methods for all db opreations and do them in another thread for example async tasks to prevent app from freezing
   * All async tasks must be statics if they are inner classes to prevent memory leaks
6. To insert some data on creating the room database we
   * Create static callback method
   * It’s return type is RoomDatabase.Callback
   * And make a new object and override oncreate method
   * Use async task to add to database and send instance as the db dao object
7. Creating A viewModel , it’s aware of activity lifecycle and responds to it
   * First extends androidViewModel not ViewModel cause AndroidViewModel Takes an application as context which will be useful to send it to repository then use it to init database
     + but on other hand if we use ViewModel we should then send activity context and if we holed a reference to an activity it’ll cause a memory leak
     + , cause viewmodel is designed to outleft the activity after it’s destroyed then we will have a refreance to destroyed activity
   * Create references to the “Repository” Class and another one to the “liveDataList” and init them in the constructor
   * Then, our activity has no access to the repository class so we create methods in the viewmodel to do repository opreations(insert,update,delete…etc)
8. Create refrance to the view model in our activity
   * We don’t use new
   * We ask android view model providers cause it knows when to create new one and when to pass it directly
   * View model provider takes
     + owner to inform the viewModel which lifecycle to follow
     + and the viewModelClass that we want object from
9. When we get a LiveData of Database entity we can set observer to it,
   * It takes the owner which is the activity cause livedata is lifecycle aware and it follows the owner
   * Then the observer class to listen to changes
   * In the onChange we can update ui the liveData will update if the activity in the foreground and not when the activity is destroyed